Analisis Performa *Container* Berplatform *Docker* atas Serangan *Malicious Software* (*Malware*)

[1]Yuliana Cahyaningrum, [2]Indrastanti Ratna Widiasari
[1,2]Fakultas Teknologi Informasi
Universitas Kristen Satya Wacana
Jl. Dr. O. Notohamidjojo 1-10, Salatiga 50715, Indonesia
Email: [1]672015031@student.uksw.edu, [2]indrastanti@uksw.edu

Masuk: 06 April 2020; Direvisi: 20 April 2020; Diterima: 28 April 2020

Abstract. *As a new virtualization technology, many things about container technology need to be explored . One of them is data security issue when this technology is applied in a network. The study aims to discover a container performance when a server is being attacked by a malware. In this research, the container is installed natively on Windows Server 2016 and using Docker as the platform. Two groups of malware are used that each group has different effect on the server system. The results show that the malware used in this research does not affect the container performance yet it affects the network used by the container. The calculation results point out an increasing delay at HTTP protocol when the server is being attacked by malware group A which is from 0.028335 ms to 2.2698161 ms. The attack of group B malware on the server caused the website inside the container inaccessible. This is because group B malware also attacked the network server where the container is holding to.*
Keywords : *Virtualization*, *Container*, *Malware*, *Native*, *Windows Server* 2016, *Docker*.

Abstrak. Sebagai teknologi virtualisasi yang baru, banyak hal yang perlu digali tentang teknologi *container*. Salah satunya adalah masalah keamanan data jika teknologi ini diterapkan dalam jaringan. Penelitian bertujuan untuk mengetahui performa *container* bila *server* mendapat serangan dari *malware*. Pada penelitian ini *container* dipasang secara *native* pada *Windows Server* 2016 dan menggunakan *Docker* sebagai *platform.* Dua kelompok *malware* digunakan dalam penelitian ini dimana setiap kelompok memiliki efek yang berbeda pada sistem *server*. Hasil menunjukkan bahwa *malware* yang digunakan dalam penelitian ini tidak mempengaruhi kinerja *container*, tetapi mempengaruhi *network* yang digunakan oleh *container*. Hasil penghitungan menunjukkan kenaikan *delay* pada protokol HTTP pada saat *server* mengalami serangan *malware* kelompok A yaitu dari 0.028335 ms sampai 2.2698161 ms. Serangan *malware* kelompok B pada *server* menyebabkan website yang ada di dalam *container* tersebut tidak dapat diakses. Hal ini disebabkan *malware* kelompok B juga menyerang *network server* dimana *container* tersebut menginduk.
Kata Kunci : Virtualisasi, *Container*, *Malware*, *Native*, *Windows Server* 2016, *Docker*.

1. Pendahuluan

Seiring dengan perkembangan jaman, teknologi IT yang ada sekarang ini diciptakan untuk semakin mempermudah kerja dan dibuat menjadi semakin efektif serta efisien secara penggunaannya serta kegunaannya. Saat ini juga teknologi IT diciptakan untuk menangani kekurangan dengan menawarkan keunggulan melebihi teknologi sebelumnya. Salah satunya adalah teknologi virtualisasi yang banyak diterapkan di perusahaan-perusahaan karena dapat menghemat pengeluaran pembelian perangkat-perangkat fisik. Hal ini disebabkan oleh konsep teknologi virtualisasi dimana beberapa *server* virtual berada dalam satu *server* fisik yang sama

[1]. Salah satu teknologi virtualisasi yang ada saat ini adalah *container*. *Container* merupakan *software* yang mengemas kode dan dependensi lain sehingga aplikasi yang berjalan dapat menjadi lebih cepat dan andal [2].

Sebagai teknologi virtualisasi yang dapat terhitung baru, banyak hal yang masih perlu digali tentang teknologi *container* ini. Salah satu contohnya adalah masalah keamanan bila diterapkan dalam jaringan. Terdapat banyak jenis serangan dalam jaringan yang dapat merusak sistem atau mengganggu kerja sistem itu sendiri. Salah satu serangan dalam jaringan adalah *Malware* atau *Malicious Software* [3]. Salah satu contoh serangan *malware* yang sangat merugikan terjadi pada tahun 2013 hingga 2017 yaitu *Trojans* dan *Malvertising*. Serangan *Ransomware* menjadi salah satu penyerang terbesar pada tahun 2017, bekerja dengan cara mengenkripsi data korban dan akan membebaskan dengan menuntut bayaran [4].

Berdasarkan permasalahan tersebut, penelitian ini dibuat untuk menganalisis performansi dari teknologi *container* dengan menggunakan *platform* aplikasi *Docker*. Hal yang akan menjadi penilaian dan analisa adalah keamanan dan kekuatan teknologi *container* tersebut untuk mengetahui dampak yang ditimbulkan setelah mendapat serangan *Malicious Software* pada sistem *server* yang ada.

2. Tinjauan Pustaka

Container merupakan salah satu hasil pengembangan teknologi terbaru yang mengusung konsep virtualisasi. Konsep dari *container* adalah membuat layanan dan sumber daya berjalan dalam *libraries, drivers* dan *binaries* yang berbeda dengan cara berbagi sebuah sistem operasi [5]. *Container-container* dapat berjalan dalam *machine* yang sama dan dapat saling berbagi kernel sistem operasi karena masing-masing *container* berjalan sebagai proses yang terisolasi. *Container* hanya akan mengisolasi *library* yang akan dijalankan saja sehingga *container* bisa menjadi lebih efektif dan ringan saat dijalankan [7].

Docker merupakan *platform* aplikasi yang menerapkan teknologi *container* di dalamnya. Perilisan *Docker container* terjadi pada tahun 2013 sebagai teknologi *open source* [7]. Konsep dari *docker* yang menjadi *platform* bagi teknologi *container* adalah *docker* berdiri diatas *hardware* dan sebuah *host operating system*. Diatas system operasi milik *host* tersebut juga dibangun *Docker engine* yang merupakan bagian dari *Docker* itu sendiri dan tempat dimana akan dijalankannya *Docker container* [8].

Selain dalam *Linux*, saat ini *container* mulai bekerja sama dengan *Microsoft* dan membuat *Docker* berjalan secara *native* sebagai *service* pada *Windows* [2]. *Docker container* sudah terdapat pada *Windows Server* 2016 dan *Windows* Desktop yaitu *Windows* 10. *Container* yang ada pada *windows server* mengikuti model *Docker* seperti yang terdapat di *Linux*, yaitu *container* yang ada pada *host* OS yang sama akan saling berbagi kernel [7]. *Windows Server Container* menyediakan isolasi bagi aplikasi melalui proses dan teknologi *namespace isolation*. *Windows server container* akan membagikan *kernel host container* dan semua *container* yang berjalan di dalam *host*. Karena *kernel* digunakan bersama, oleh karena itu semua *container* yang ada di dalam *kernel* tersebut memerlukan versi dan konfigurasi kernel yang sama [9].

Terdapat penelitian berupa penerapan teknologi *container* dengan *Docker* untuk pengelolaan aplikasi-aplikasi *web* melalui studi kasus Jurusan Teknik Informatika UNESA. Penelitian tersebut membahas tentang teknologi *container* berplatform *Docker* yang digunakan sebagai solusi untuk mengelola aplikasi *web* yang terdapat di jurusan Teknik Informatika Universitas Negeri Surabaya. Penggunaan *Docker* semakin mempermudah proses *deployment* aplikasi *web* beserta *software* pendukung, seperti *web server*, *database server*, dependensi, dan *environment* lain ke *server* sehingga memberikan solusi pada aplikasi *web* yang membutuhkan *Docker* untuk bereksperimen [1].

Penggunaan teknologi *container* juga diterapkan pada penelitian lain yang berisi tentang pengimplementasian *Virtual Data Center* dengan menggunakan *Container* berbasis *Docker* dan SDN pada sistem operasi *Linux*. Penelitian tersebut dilakukan untuk menjawab permintaan akses yang dinamis dan *on-demand* berdasarkan teori *Software Defined Networking* (SDN) yang

merupakan teknik jaringan berbasis aplikasi yang menyebabkan pengguna bisa langsung menjalankan. *Software* tersebut akan menangani konfigurasi yang dibutuhkan dalam jaringan. Terlebih ini akan sangat fleksibel jika diterapkan dalam jaringan *Data Center*. Pada penelitian tersebut, *Docker* yang diterapkan digunakan sebagai *host* dari *Data Center* berbasis virtual yang dapat terintegrasi dengan baik ke dalam *switch* jaringan *Data Center* yang dikontrol dengan *controller* berbasis teknologi SDN [6].

Malicious Software dikenal sebagai program yang berbahaya bagi sistem. Ada beberapa tipe *malware* yaitu *spyware*, *adware*, *phising*, virus, *trojan horse*, *worm*, dan *ransomware*. *Malware* bisa menyerang sebuah perangkat lewat akses internet, *email*, saat sedang mengakses sebuah *website,* demo *game*, pemasangan sebuah aplikasi, dan saat mendownload *file* dari internet [10].

Salah satu penilaian dalam kualitas layanan sebuah jaringan adalah *delay* yang terjadi dalam jaringan tersebut. *Delay* merupakan lama penundaan waktu yang ditempuh dalam proses transmisi data dari titik awal ke titik tujuan [11]. Pada dasarnya *delay* merupakan hal yang sangat menentukan kualitas layanan. Untuk mendapatkan hasil rata-rata *delay* yang ada dalam jaringan, perhitungan menggunakan persamaan 1 [12].

$$delay\ rata - rata = \frac{total\ delay}{total\ paket} \quad (1)$$

Dimana:
Total delay = jumlah dari keseluruhan delay 1, 2, 3, … , n.
Total paket = jumlah dari paket transmisi data yang diterima

Fiddin dkk melakukan pengukuran performansi *container* menggunakan *platform Docker* pada saat mengalami serangan DoS TCP SYN. Penelitian tersebut bertujuan untuk menganalisis kinerja mesin dari sisi *overall performance*, dan layanan *web server* yang dijalankan di atas *container* pada saat keadaan normal dan saat mendapatkan serangan. Pengujian dilakukan dengan membandingkan performansi pada mesin *native*. Hasil yang didapat adalah serangan DoS memberikan dampak penurunan performansi dari sisi *overall performance* dan layanan *web server*, pada mesin *native* dan pada *container Docker*. Pada serangan yang relatif ringan didapatkan hasil parameter *request* per *second* yang menyebabkan penurunan performansi pada *native* sebesar 40.22% dan 37.65% pada *Docker*. Sedangkan pada parameter *response time web server*, serangan DoS menyebabkan peningkatan *response tim*e pada *server* native sebesar 40.80%, sedangkan pada *Docker* sebesar 38.38% [5].

3. Metodologi Penelitian

Berdasarkan penelitian yang dilakukan yaitu tentang Analisis Performansi Virtualisasi *Container* dengan *Docker* atas Serangan *Malicious Software* (*Malware*), terdapat beberapa langkah yang digunakan agar bisa mendapatkan hasil yang sesuai. Pada tahap awal dalam penelitian ini, dilakukan perumusan masalah yang berisi tentang rumusan dari apa saja masalah yang terjadi sehingga dibuatlah penelitian ini. Langkah selanjutnya adalah studi literatur, dilakukan untuk digunakan untuk dijadikan referensi teori dalam penelitian. Referensi yang digunakan merupakan referensi teori yang berkaitan dan relevan dengan kasus dan masalah dalam penelitian. Tahap ketiga adalah menganalisis kebutuhan yang terkait dengan penelitian yang dilakukan berupa *hardware* dan *software* yang digunakan. Dalam penelitian ini, spesifikasi *hardware* dan *software* yang digunakan oleh PC *server* dapat dilihat pada Tabel 1.

Tabel 1. *Spesifikasi PC Server*

No.	Nama	Spesifikasi
1	Processor	Intel(R) Core(TM) i3-7100 CPU @ 3.90GHz (4 CPUs), ~3.90GHz
2	RAM	4096MB
3	Sistem Operasi	Windows *Server* 2016 Datacenter Evaluation

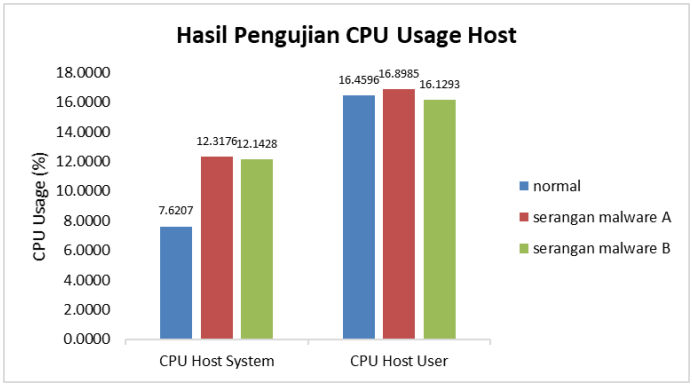
Perangkat lain yang digunakan dalam penelitian ini adalah *Switch* TP-*Link* TL-SF1008D yang digunakan sebagai penghubung antara PC *Server* dengan *client*. Dalam penelitian ini,

digunakan 2 pc *client*. Tahap selanjutnya adalah tahap implementasi dimana merupakan proses mengubah kebutuhan yang ada dalam tahap perancangan menjadi sebuah sistem yang diimplementasikan secara nyata. Pada tahap keenam, hal yang dilakukan adalah pengujian sistem yang sudah dibuat. Pengujian dilakukan dengan menerapkan tiga skenario yang berbeda. Pengujian yang pertama adalah menguji performa pada pc *server* yaitu pada *container* dalam saat sistem dalam keadaan normal. Pengujian kedua dan ketiga adalah menguji performa *container* pada saat *server* terserang dua kelompok *malware* yang berbeda. Tahap ketujuh adalah penganalisisan hasil dari pengujian yang sudah dilakukan. Analisa yang dilakukan dengan memperhatikan hasil dan memperhatikan hasil dari kondisi *server*, *container*, dan *website* sebagai layanan yang disediakan oleh *container* bagi *client*. Tahap terakhir yang dilakukan adalah menarik kesimpulan. Dengan mendapatkan hasil analisis pengujian yang sudah dilakukan, maka didapatkan kesimpulan tentang pengujian yang dilakukan.

4. Hasil dan Diskusi

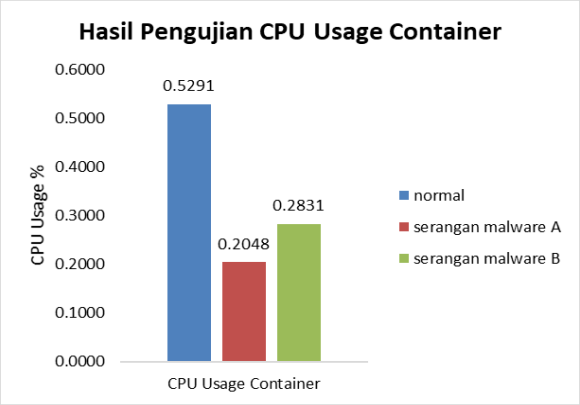
Berdasarkan dengan metodologi yang telah dibuat dalam penelitian ini, *container* berplatform *Docker* dijalankan secara *native* yaitu berada langsung diatas Windows *Server* 2016 dan akan diakses oleh dua *client*. *Client* mengakses layanan yang ada di dalam *container* yaitu *website*. Aplikasi yang digunakan untuk memantau kondisi *server* dan *container* yang berjalan adalah *Splunk Enterprise*. Aplikasi *Splunk Enterprise* dapat digunakan untuk mengumpulkan, menganalisis, dan melakukan tindakan berdasarkan data serta hasil yang didapatkan pada teknologi infrastruktur, sistem keamanan, dan aplikasi bisnis yang dimiliki [13]. Jenis *malware* yang digunakan dalam penelitian ini akan dibagi menjadi dua kelompok, yaitu kelompok A yang terdiri dari *adware*, *grayware*, *riskware*, dan PUP (*Potentially Unwanted Program*). Sedangkan kelompok B terdiri dari *adware*, *grayware*, *riskware*, *spyware*, *worm*, *trojan*, dan PUP (*Potentially Unwanted Program)*. Meskipun kedua kelompok *malware* tersebut memiliki persamaan dari jenis yang digunakan, namun nama macam *malware* yang digunakan berbeda. Sehingga setiap *malware* yang digunakan memiliki daya rusak yang berbeda-beda. Kedua malware tersebut menyerang sistem-sistem komputer dengan memberatkan kinerja komputer, mengganggu bagian jaringan, dan memperberat memori komputer. Hal lain yang perlu digarisbawahi dari penelitian ini adalah besar hasil yang didapat bukan hasil mutlak bagi pengujian lain. Hal ini dikarenakan hasil pengujian yang dapat berbeda dipengaruhi oleh berbagai factor. Beberapa diantaranya adalah penggunaan *hardware* dan *software* komputer *server* yang berbeda, banyaknya aplikasi yang berjalan pada komputer *server* tersebut, dan penggunaan *malware* dengan jenis yang sama.

Pengujian pertama yang dilakukan adalah pengukuran CPU *usage* dari *host* dan *container*. Pengukuran CPU Usage dimaksudkan untuk mengetahui tinggi penggunaan CPU dari setiap pengujian. Pengujian-pengujian tersebut dilakukan dengan batasan waktu tertentu, yaitu selama 40 menit, dan dilakukan sebanyak tiga kali pengujian. Dari tiga hasil pengujian, dirata-rata dan dijadikan nilai akhir dalam pengujian ini. Pengujian dilakukan sebanyak tiga kali karena dari tiga pengujian tersebut sudah menunjukkan perbedaan yang cukup signifikan.



Gambar 1. Diagram hasil pengujian CPU *Usage* pada *Host*

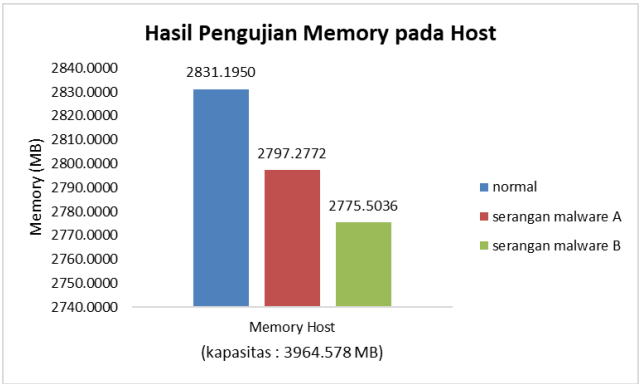
Gambar 1 menunjukkan hasil pengujian CPU *usage* pada *host.* Setelah melakukan pengujian, didapatkan rata-rata dari hasil pengujian yaitu CPU *System* milik *host* berada pada 7.6207% saat normal, 12.3176% saat mendapat serangan dari kelompok *malware* A, dan 12.1428% saat *server* mendapat serangan dari kelompok *malware* B. Sedangkan pada CPU *User* yang terletak pada *host,* saat keadaan normal 16.4596%, saat mengalami serangan *malware* kelompok A menjadi 16.8985%, dan 16.1293% saat *host server* mendapat serangan dari *malware* B. Berdasarkan hasil tersebut, kenaikan CPU *Usage* yang cukup signifikan terjadi pada *system host*. Hal itu disebabkan karena *malware-malware* yang digunakan dalam penelitian ini bekerja dengan menyerang sistem, dan bekerja di dalam sistem.



Gambar 2. Diagram hasil pengujian CPU *Usage* pada *Container*

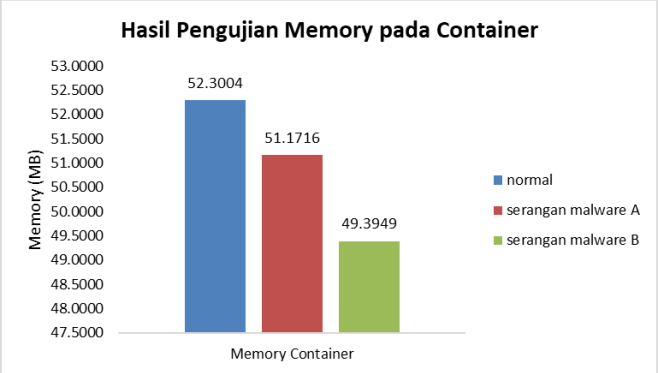
Seperti yang ditunjukkan pada Gambar 2, hasil rata-rata yang didapat dari pemantauan CPU *Usage* pada *container* pada saat *server* dalam keadaan normal berada pada nilai 0.5291%, saat *server* mengalami serangan *malware* kelompok A menjadi 0.2048%, dan saat mengalami serangan *malware* kelompok B adalah 0.2831%. Tingginya penggunaan CPU pada *container* dipengaruhi oleh banyaknya *client* yang mengakses layanan di dalam *container* tersebut, dan seberapa banyak layanan di dalam *container* itu diakses. Semakin banyak pengaksesan yang terjadi, maka CPU *Usage* dari *container* akan naik. Saat penelitian, pada keadaan *server* normal, *container* diakses berulang kali, dan dapat bekerja dengan baik sehingga CPU *Usage* menjadi tinggi. Meskipun *container* tetap berjalan saat *server* mengalami serangan dari *malware* A dan B, layanan yang ada di dalam *container* mengalami kendala sehingga berpengaruh juga terhadap tingginya CPU *Usage* dari *container* tersebut.

Pengujian selanjutnya adalah pengukuran *memory* milik *host* dan *container* dilakukan dengan batasan waktu selama 40 menit dan dilakukan tiga kali pengujian. Pengukuran *memory* dimaksudkan untuk mengetahui perbedaan *memory server* sebelum dan setelah adanya serangan malware mengingat *malware* yang digunakan dalam penelitian juga diketahui dapat mempengaruhi kinerja *memory* komputer. Hasil yang didapat kemudian diambil nilai rata-rata dan mendapat hasil seperti yang ditunjukkan pada Gambar 3 dan Gambar 4.



Gambar 3. Diagram hasil pengujian *Memory Host*

Gambar 3 menunjukkan PC *host* memiliki kapasitas *memory* maksimal sebanyak 3964.578 MB. Hasil monitoring pada *memory* saat *server* dalam keadaan normal, *host* menggunakan *memory* sebesar 2831.195041 MB. Namun pada saat *host* mengalami serangan *malware* kelompok A, terjadi sedikit penurunan penggunaan *memory* oleh *host* sebanyak 33.9178 MB sehingga menjadi 2797.2772 MB. Sementara, dengan serangan *malware* kelompok B adalah 2775.5036 MB. Berdasarkan hasil tersebut, *malware-malware* yang digunakan dalam penyerangan *server* tidak mempengaruhi *host* secara signifikan dalam penggunaan *memory*.



Gambar 4. Diagram hasil pengujian *Memory Container*

Gambar 4 menunjukkan pada saat keadaan normal, *container* menggunakan 52.3004 MB, dan mengalami penurunan sebesar 1.1288 MB saat terjadi serangan *malware* kelompok A pada *host* menjadi 51.1716 MB, serta menjadi 49.3949 MB saat *malware* kelompok B menyerang *server*. Banyaknya penggunaan *memory* pada *container* dipengaruhi oleh tingginya permintaan layanan yang masuk pada *container* tersebut. Mengingat terdapat kendala yang mempengaruhi kinerja *container* pada saat *malware* menyerang mengakibatkan tidak banyak layanan yang dapat dilayani oleh *container*.

Sebagai teknologi virtual, *container* tetap membutuhkan IP untuk bertukar data atau melakukan transmisi data. Karena *container* berjalan dengan menginduk kernel dari sistem operasi pada *server*, *container* menginduk juga pada *network interfaces* yang terdapat pada *server* tersebut agar *container* tersebut dapat diakses atau mengakses jaringan eksternal *server*. Pada penelitian ini, *network* milik *container* menginduk pada salah satu *Ethernet network* yang ada pada pc *server*.

Pengujian lain adalah memantau *website* yang berperan sebagai layanan yang disediakan *container* bagi *client* yang dilakukan menggunakan aplikasi *Wireshark* dengan memantau protokol-protokol yang berjalan saat mengakses *website* tersebut. Dengan memantau website tersebut, akan diketahui perubahan yang terjadi saat pengaksesan website dilakukan. Hasil yang diambil dalam penelitian ini berasal dari *capture* jaringan pada *Wireshark* dengan memfilter protokol HTTP.

Pengujian dengan *Wireshark* yang pertama dilakukan saat *server* dalam keadaan normal. Hasil yang didapat dalam kondisi ini adalah *website* dapat terbuka dengan lancar. Untuk membuktikannya dilakukan penghitungan *delay* pada protokol HTTP milik *website* tersebut. Dengan melakukan penghitungan menggunakan persamaan 1, maka hasil yang didapat adalah sebagai berikut :

$$delay\ rata-rata = \frac{total\ delay}{total\ paket} = \frac{0.005667\ s}{20} = 0.00028335\ s$$

Pengujian juga dilakukan dengan cara mengakses *website*, lalu melakukan *capture* pada jaringan yang digunakan oleh *container* saat *server* mendapat serangan dari *malware* kelompok A. Hasil yang didapat adalah *website* dapat diakses namun membutuhkan waktu yang lebih lama hingga halaman *website* dapat tampil. Pembuktian dilakukan dengan penghitungan *delay* protokol HTTP menggunakan persamaan 1, maka hasil yang didapat sebagai berikut :

$$delay\ rata - rata = \frac{total\ delay}{total\ paket} = \frac{0.703643\ s}{31} = 0.022698161\ s$$

Jika hasil tersebut dibandingkan dengan hasil saat *server* dalam keadaan *normal*, maka terjadi kenaikan waktu *delay* selama 2.2414811 ms. Hasil tersebut menunjukkan terjadinya waktu *delay* yang meningkat secara signifikan saat *server* mendapat serangan *malware*.

Pengujian pada website selanjutnya yang dilakukan sama seperti dengan pengujian sebelumnya namun menggunakan serangan dari kelompok malware B. Hasil yang didapat adalah *website* tidak dapat diakses dan terbuka. Saat dilakukan pengujian dengan *capture* menggunakan *Wireshark* pada jaringan milik *container*, tidak terdapat hasil yang menunjukkan bahwa ada protokol HTTP yang tertangkap. Hal ini dikarenakan *malware* kelompok B bekerja dengan mengganggu *interface network* yang ada pada *server*. Hal ini dibuktikan dengan dilakukannya pengiriman paket *ping* dari *server* dan menggunakan alamat tujuan IP milik *container* tersebut. Hasil yang didapat adalah *Request Time Out* pada semua pengiriman paket. Hasil *capture* pada jaringan juga tidak dapat menangkap protokol ICMP yang berjalan. Berdasarkan hal tersebut, dapat diketahui bahwa *network* yang digunakan *container* terganggu karena adanya serangan *malware* pada *server* dan menjadi tidak dapat diakses meskipun *container* tetap berjalan.

Dengan konsep teknologi *container* yang berjalan sebagai proses yang terisolasi, hasil-hasil yang didapat dari pengujian yang telah dilakukan semakin memperjelas konsep tersebut. Hal ini dibuktikan dari pengujian *memory* dan CPU *Usage* milik *container* yang menurun saat terjadi penyerangan kelompok *malware* A dan B dibandingkan saat *server* dalam kondisi normal, karena kedua kelompok *malware* tersebut mengganggu kinerja jaringan pada *server*. Dengan adanya gangguan kinerja pada jaringan *server*, maka *client* menjadi terganggu saat mengakses layanan yang ada di *container*. Sedangkan tingginya CPU *Usage* dan *memory container* dipengaruhi dari tingginya pengaksesan layanan yang ada di dalam *container* tersebut.

5. Simpulan dan Saran

Berikut beberapa kesimpulan yang dapat ditarik berdasarkan hasil penelitian yang telah dilakukan. Tingginya hasil CPU *Usage* yang terjadi pada *host* terpengaruh dengan banyaknya aplikasi atau layanan yang sedang bekerja di dalamnya. Berdasarkan hasil pengujian pada CPU *Usage* pada *host*/*server*, serangan *malware* kelompok A maupun kelompok B mempunyai pengaruh yang cukup besar terhadap tingginya CPU *Usage* terlebih pada sistem *host*, karena *malware* yang digunakan dalam penelitian ini berjalan pada bagian sistem *host*. Setiap kelompok *malware* yang digunakan saat penyerangan memiliki efek yang berbeda-beda terhadap terhadap sistem pada *server*, karena ada *malware* yang hanya mengganggu kinerja hingga merusak sistem pada *server*. *Malware* yang digunakan dari penelitian ini tidak banyak mempengaruhi penggunaan *memory* pada *host*, sehingga tidak terjadi selisih yang signifikan dari saat *server* dalam keadaan normal dengan *server* yang mengalami serangan dari *malware* kelompok A maupun *malware* kelompok B.

Berdasarkan penelitian yang telah dilakukan terhadap *container*, tingginya CPU *Usage* dan *memory* pada *host* tidak mempengaruhi tinggi CPU *Usage* dan memory milik *container*. *Malware* yang digunakan dalam penelitian ini tidak mempengaruhi kinerja dari *container*, tetapi mempengaruhi *network* yang digunakan oleh *container*. Berdasarkan hasil penghitungan *delay* pada protokol HTTP pada saat *server* mengalami serangan *malware* kelompok A, terjadi kenaikan *delay* menjadi 2.2698161 ms dari 0.028335 ms saat *server* dalam keadaan normal. Dan saat terjadi serangan *malware* kelompok B pada *server*, menyebabkan *website* yang ada di dalam *container* tersebut menjadi tidak dapat diakses karena malware kelompok B juga menyerang *network server* dimana *container* tersebut menginduk.

Referensi

[1] F. Adiputra, Container dan docker : teknik virtualisasi dalam pengelolaan banyak aplikasi web, *J. Ilm. SimanteC*, vol. 4(3), 2015.

[2] Docker Inc, "What is a Container," *Docker Inc*, 2018. [Online]. Available: https://www.docker.com/resources/what-container. [Accessed: 27-Nov-2018].
[3] N. Budhisantosa, Analisis modifikasi konfigurasi access control list pada usb flash disk studi kasus pada penyebaran malware trojan shortcute, *J. Ilmu Komput.*, vol. 10, pp. 60–71, 2014.
[4] Malwarebytes, "Malware," *Malwarebytes*, 2018. [Online]. Available: https://www.malwarebytes.com/malware/. [Accessed: 23-Nov-2018].
[5] C. Fiddin, R. Mayasari, and R. Munadi, Analisis performansi virtualisasi container menggunakan docker dibawah serangan networked denial of service, *e-Proceeding Eng.*, vol. 5(1), pp. 281–290, 2018.
[6] M. F. Alauddin, R. M. Ijtihadie, and M. Husni, Implementasi virtual data center menggunakan linux container berbasis docker dan SDN, *J. Tek. ITS*, vol. 6(2), pp. 6–8, 2018.
[7] Docker Inc, "Docker Windows Containers," *Docker Inc*, 2018. [Online]. Available: https://www.docker.com/products/windows-containers. [Accessed: 22-Nov-2018].
[8] Docker Inc, "About Docker Engine," *Docker Inc*, 2018. [Online]. Available: https://docs.docker.com/engine/. [Accessed: 22-Nov-2018].
[9] Microsoft, "Containers on Windows," *Microsoft*, 2016. [Online]. Available: https://docs.microsoft.com/en-us/virtualization/windowscontainers/. [Accessed: 22-Nov-2018].
[10] AVAST Software Inc, "Malware," *AVAST Software, Inc*, 2018. [Online]. Available: https://www.avast.com/c-malware. [Accessed: 27-Nov-2018].
[11] H. Fahmi, Analisis qos (quality of service) pengukuran delay, jittes, packet lost, dan throughput untuk mendapatkan kualitas kerja radio streaming yang baik, *J. Teknol. Inf. Dan Komun.*, vol. 7(2), pp. 98–105, 2018.
[12] R. Oktavianus Lukas Sihombing and M. Zulfin, Analisis kinerja trafik web browser dengan wireshark network protocol analizer pada sistem client-*server*, *Singuda Ensikom*, vol. 2(3), pp. 96–101, 2003.
[13] Splunk, "Splunk Enterprise 7.2.5.1," *Splunk*, 2019. [Online]. Available: https://www.splunk.com/en_us/download/splunk-enterprise.html. [Accessed: 23-Nov-2018].